Lab 5 Report

1st Skyler Cook
Electrical and Computer Engineering
Utah State University
Logan, USA
A02304185@aggies.usu.edu

2nd Jaron Seedal
Electrical and Computer Engineering
Utah State University
Logan, USA
a02358703@aggies.usu.edu

3rd Dr. Jonathan Philips
Electrical and Computer Engineering
Utah State University
Logan, USA
jonathan.phillips@usu.edu

***Abstract*—This document reports the laboratory work performed to implement a 24-bit accumulator on the DE10-Lite FPGA development board using a VHDL finite-state machine (FSM). The accumulator accepts a 10-bit unsigned input from the board's toggle switches and displays the 24-bit accumulated sum in hexadecimal across the six 7-segment displays. One push button serves as an active-low reset (clears accumulator to zero), and the other push button triggers an accumulate operation that adds the current switch value to the stored sum.**

I. Introduction

This project implements a 24-bit accumulator on the DE10-Lite development board using VHDL. The accumulator stores a 24-bit unsigned sum that is updated when the user sets the 10 toggle switches to a desired value and presses the 'add' button. The design meets the following requirements: reflect the switch values on the ten LEDs, set the accumulator to all 'zeroes' with the reset button, display the accumulator as six hex digits on the seven-segment displays, and guarantee exactly one accumulation per physical press of the add button by debouncing the 'add' input. A finite-state machine (FSM) was used to implement debouncing.The implementation was tested on hardware to confirm correct behavior.

II. Objectives

The objectives of the lab were:

1. Implement the accumulator as a synchronous VHDL FSM on the DE10-Lite board.

2. Use one push button as the asynchronous reset input (active low) to clear the accumulator to zero.

3. Use the other push button as an 'add' control: when pressed, add the 10-bit value present on the board switches to the 24-bit accumulator.

4. Mirror the 10 switches on the 10 LEDs so each switch's logic state is visible.

5. Display the 24-bit accumulated value in hexadecimal across the six 7-segment displays.

6. Debounce the add button so that one physical press results in exactly one accumulation operation.

7. Validate the design through correct operation on the DE10-Lite.

III. Procedure

The design process for the 24-bit accumulator followed a logical development sequence beginning with planning and ending with successful hardware implementation.

1. FSM Design and Planning

The first step was to outline the finite state machine (FSM) that would control the accumulator's operation. The FSM was drawn on paper to visualize the three main states: IDLE, ADD, and WAITRELEASE. Each transition was defined based on the add button input and the debounce counter behavior. This preliminary chart served as the blueprint for the control logic. The finalized FSM state transition diagram is shown in Figure 1.

2. Initial VHDL Development

A rough draft of the VHDL code was written entirely from scratch. In prior labs, the students used classroom examples as code templates. This project was implemented without a reference design. As a result, early compilation attempts produced numerous syntax errors. These issues were gradually resolved through careful line-by-line debugging, reinforcing a better understanding of how to structure VHDL architectures, components, and signal assignments from the ground up.

3. Testbench Creation and Debugging

A testbench was written to simulate push-button presses, switch inputs, and reset behavior. Initial simulations did not behave as expected because of logic errors and incorrect process triggers. To debug these errors, the code was sunthesized in Quartus without pushing to the board to see if any more errors would show. In doing this, the team discovered that a register was being driven in multiple processes, and was corrected.

4. Debounce Logic Correction

During simulation, it became clear that the debounce counter wasn't behaving as anticipated. The counter was redesigned to operate synchronously with the 50 MHz system clock so that the add button input would be properly timed and stable. After this fix, the simulated FSM transitions occurred

exactly once per button press, demonstrating that the debounce and one-shot behavior were working correctly.

5. Hardware Implementation and Testing

Once the simulation results were verified, the design was compiled and uploaded to the DE10-Lite board. On the very first upload, the circuit performed flawlessly. The LEDs mirrored the switch positions, the accumulator updated exactly once per button press, and the hexadecimal display showed the correct running total. Both the reset and add buttons responded immediately and consistently, confirming that the FSM and debounce logic operated as intended.

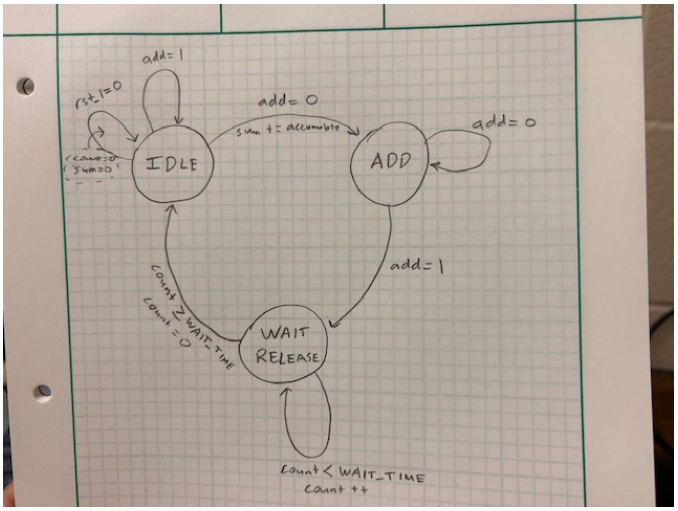

Fig. 1. State Transition Diagram

IV. Results

All project objectives were successfully met. The simulation showed that all values were reset when the button was pressed, and that if the add signal went low then a single instance of the switch signal would be added and put to the hex values. However, the switch value would not be added again even in the add signal went low until the add signal went high for a relatively large amount of time. This is shown in Figure 2 and Figure 3.

The accumulator correctly added the 10-bit switch input to the running 24-bit total each time the add button was pressed, with the value displayed in hexadecimal across the six 7-segment displays. The LEDs accurately reflected the state of the switches, and the reset button reliably cleared the accumulator to zero. The debounce logic functioned as intended, producing exactly one accumulation per button press. The design operated correctly on the DE10-Lite board without further modification after the initial upload.

V. Discussion and Conclusions

This project reinforced the importance of careful FSM planning and the value of a structured, step-by-step debugging process. Writing the design entirely from scratch provided

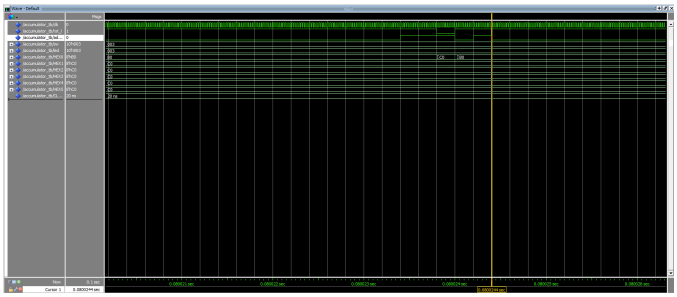

Fig. 2. Values resetting with Reset Signal

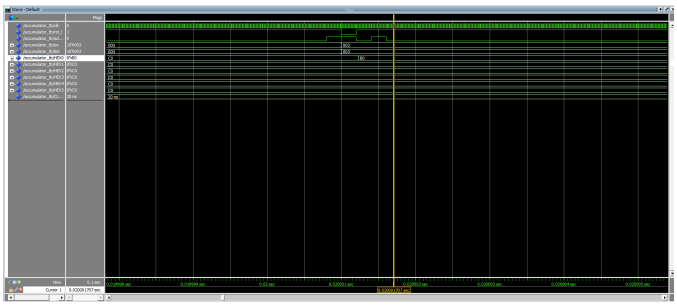

Fig. 3. Adding with Debouncing

hands-on experience with VHDL syntax, process sensitivity, and signal timing. Implementing a synchronous debounce counter highlighted the need to align all logic with the system clock for reliable hardware performance. Overall, this lab strengthened understanding of FSM design, code debugging, and development flow for hardware implementation.